Analytik NEWS
Das Online-Labormagazin

analytik.news Publikationsdatum: 05.09.2019

Henderson-Hasselbalch forever? – Puffer berechnen im Zeitalter der Digitalisierung

Dr. Michael Luthardt

Jeder, der Chemie als Beruf erlernt, sei es in einer Ausbildung oder im Studium, lernt die Henderson-Hasselbalch-Gleichung

$$pH = pK_a + log\frac{[Anion]}{[Säure]} \quad (1a)$$

zur Berechnung des pH-Wertes einer Pufferlösung kennen. [Anion] und [Säure] sind die Konzentrationen einer schwachen Säure und ihres Salzes im Gleichgewicht, pK_a die logarithmierte Säurekonstante. Ersetzen wir [Anion] und [Säure] durch die analytischen Konzentrationen des Salzes c_0(MA) und der Säure c_0(HA) so wird aus Gleichung (1a)

$$pH = pK_a + log\frac{c_0(MA)}{c_0(HA)}. \quad (1b)$$

Für gleiche Konzentrationen c_0(MA) und c_0(HA) ist der pH = pK_a. Mit dieser Gleichung endet die Berechnung des pH einer Pufferlösung in Lehrbüchern (z. B. [1], [2], [3]).

Es wird nicht darauf hingewiesen, dass es sich hier um eine sehr eingeschränkte Näherung handelt, und statt c_0(MA) findet man zumeist $c_0(A^-)$ oder c_0(Base). Auch das ist nicht ganz korrekt.

Natürlich muss richtigerweise mit molalen Aktivitäten anstatt mit Konzentrationen gerechnet werden. Im sauren/neutralen pH-Bereich und nicht zu hohen Ionenstärken ist die Aktivitätskorrektur gering. Ebenso rechnen wir hier nur mit 10^{-14} mol²/L² für das Ionenprodukt des Wassers. Diese Korrekturen treten gegenüber den Überlegungen, um die es hier gehen soll, zurück.

Leicht sieht man, dass die Gleichung (1) Grenzen haben muss: der pH wäre von den absoluten Konzentrationen unabhängig. Mit zunehmender Verdünnung muss der pH jeder Pufferlösung jedoch gegen 7 gehen. Eine weitere Beschränkung ist, dass der pK_a nicht zu stark von 7 abweichen darf. In beiden Fällen liefert Gleichung (1) von den tatsächlichen Verhältnissen abweichende Werte, weil die Gleichgewichtskonzentrationen von [Anion] und [Säure] nicht durch die analytischen Konzentrationen ersetzt werden können, sondern aus den simultanen Säure-Base-Gleichgewichten berechnet werden müssen.

Natürlich finden sich Monographien und Fachartikel (z. B. [4], [5], [6]), die dies thematisieren und exakte Lösungen geben. Aber warum haben exakte Lösungen trotz allem Digitalisierungshype so wenig Eingang in die Sekundärliteratur gefunden?

Zum einen fallen in der Praxis diese Abweichungen nicht auf, weil meist

1. der pK_a für Pufferlösungen nicht zu weit von 7 entfernt ist,
2. Konzentrationen um 0,1 M gewählt und
3. heute Fertigpuffer gekauft werden, auf deren pH man sich verlässt (und dies auch mit gutem Gewissen kann).

Natürlich sind Fertigpuffer fast immer Mehrkomponentengemische, deren exakte pH-Berechnung extrem aufwändig wäre. Dennoch ist es für Verständnis und Veranschaulichung des Themas in der Ausbildung hilfreich und sinnvoll, grundlegende Abhängigkeiten über die Gleichung (1) hinaus zu diskutieren.

Für Ausbildung und Studium scheint hinzu zu kommen, dass exakte pH-Berechnungen sehr schnell auf Gleichungen höheren Grades in $[H^+]$ führen. Die Berechnung beginnt für reines Wasser quadratisch,

$$[H^+]^2 - K_W = 0.$$

Jedes weitere Gleichgewicht, an dem Wasserstoff-Ionen beteiligt sind, erhöht den Grad des Polynoms um 1. So müssen wir für die Berechnung des pH der Lösung einer einbasischen Säure bereits eine Gleichung 3. Grades in $[H^+]$ lösen und für den exakten pH einer zweibasischen Schwefelsäure im analytisch interessanten Konzentrationsbereich um 0,1 M bereits ein Polynom 4. Grades.

Einerseits lassen sich Gleichungen dritten und vierten Grades geschlossen lösen [7]. Dabei kommt uns entgegen, dass ein pH-Wert Problem – mathematisch richtig formuliert – genau eine reelle positive Lösung hat.

Andererseits ist die Berechnung der Nullstellen höherer Polynome mit Papier und Bleistift nichts, was man routinemäßig durchführen möchte. Schlimmstenfalls müssen alle Nullstellen berechnet werden, wenn erst das letzte Ergebnis die gesuchte reelle positive Lösung liefert. Verständlich, dass in der ferneren Vergangenheit hierzu keine große Neigung bestand. Jedoch hätte sich dies mit der seit über dreißig Jahren flächendeckenden Verbreitung hinreichend leistungsfähiger Rechentechnik ändern können, ist aber nicht geschehen. Dabei ist dies ein Thema, in dem man chemische Grundausbildung zwanglos mit allgemein anwendbarem mathematischem Wissen und Programmierkenntnissen nutzbringend verbinden kann.

Die pH-Berechnung einer Pufferlösung

Für die exakte Berechnung der Wasserstoffionenkonzentration in einer Pufferlösung gehen wir von folgenden Reaktionsgleichungen aus:

$H_2O \rightleftarrows H^+ + OH^-$ (a)

$HA \rightleftarrows H^+ + A^-$ (b)

$MA \rightarrow M^+ + A^-$ (c)

Die Konzentrationen $[H^+]$, $[OH^-]$, $[HA]$, $[A^-]$ und $[M^+]$ im Gleichgewicht sind die 5 Unbekannten in diesem System, zu deren Berechnung wir 5 unabhängige mathematische Gleichungen benötigen. Diese liefern uns die chemischen Gleichgewichte

$[H^+] \cdot [OH^-] = K_W$ (2-1)

$[H^+] \cdot [A^-] / [HA] = K_a$, (2-2)

die Massebilanzen

$[M^+] = c_0(MA)$ (2-3)

$[HA] + [A^-] = c_0(HA) + c_0(MA)$ (2-4)

und die Elektroneutralitätsbedingung

$[H^+] + [M^+] = [OH^-] + [A^-]$ (2-5)

c_0 sind die analytischen Konzentrationen des Salzes MA und der schwachen Säure HA.

Sind dies einwertige Kationen und Anionen, können wir vollständige Dissoziation und damit Gleichung (2-3) voraus setzen.

Der entscheidende Punkt, um die Charlot-Gleichung herzuleiten, ist Gleichung (2-4). $[A^-]$ ist nicht $c_0(MA)$; wir müssen demgegenüber mit Gleichung (2-4) die Gesamtbilanz der Anionen aufstellen.

Diese finden sich entweder frei oder in der Säure gebunden. K_W und K_a sind das Ionenprodukt des Wassers bzw. die Säurekonstante der schwachen Säure.

Einsetzen und zusammenfassen führt auf die kubische Gleichung (3):

$$[H^+]^3 + (c_0(MA) + K_a)\,[H^+]^2 - (c_0(HA)\,K_a + K_W)\,[H^+] - K_a \cdot K_W = 0 \quad (3)$$

Diese Berechnung wurde zuerst 1947 von Gaston Charlot in etwas anderer Form

$$[H^+] = K_a \cdot \frac{c_0(HA) - \Delta}{c_0(MA) + \Delta} \quad (4)$$

mit $\Delta = [H^+] - [OH^-]$ angegeben [8] und wird in der englischsprachigen Literatur als Charlot-Gleichung bezeichnet.

Eine Lösung von Gleichung (3) lässt sich für gegebene Parameter mit Mathematikprogrammen wie dem freien GNU Octave [9] leicht erhalten.

Octave gibt es mit oder ohne grafische Benutzeroberfläche für alle wichtigen Betriebssysteme. Wer sich selber an der Programmierung der Lösung von Polynomen bis zum 4. Grad versuchen möchte, dem sei GNU bc [10] empfohlen. Es gibt auch eine Windows-Version und die selbst erstellten Programme kann man über VB-Skripte direkt, ohne das Windows Terminalfenster benutzen zu müssen, aufrufen.

Im Folgenden benutzen wir Octave. Hier lassen sich mit Grundkenntnissen der Programmierung sehr leicht numerische Lösungen für die Nullstellen beliebiger Polynome erhalten. So sind noch weit komplexere pH-Berechnungen einfach möglich und Ergebnisse lassen sich grafisch darstellen.

GNU Octave ist weitgehend kompatibel mit dem kommerziellen Paket Mathematica.

In den meisten Linux-Distributionen lässt sich Octave aus den Repositories installieren. Beim ersten Start sollte man als Standard-Arbeitsverzeichnis für Octave den Ordner festlegen, in dem man seine Programme speichern möchte.

Die beiden wichtigsten Reiter am unteren Rand des Fensters sind das Befehlsfenster und der Editor. Unter Dokumentation steht das vollständige Handbuch in Englisch zur Verfügung; zur Einarbeitung ist es besser, eine der zahlreichen freien Darstellungen wie [9] zu benutzen.

Octave unterscheidet dateimäßig zwischen Funktionen und Skripten. Es ist zweckmäßig, wiederkehrende Berechnungen mit Variablen als Funktionen `func.m` im Arbeitsverzeichnis zu speichern.

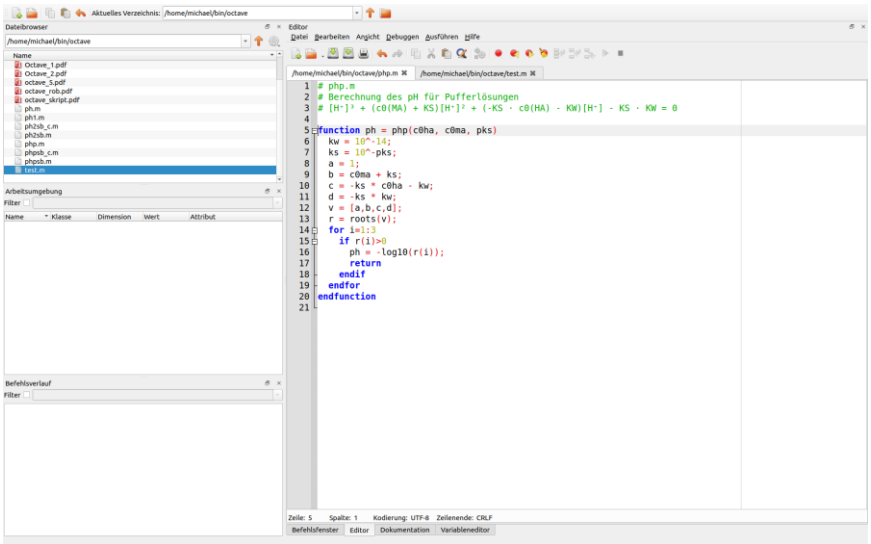

Abb. 1: Screenshot GNU Octave

Für die Lösung der Gleichung (3) lautet der Programmcode der Funktion php wie in Abbildung 2 dargestellt.

Funktionsname, hier php, und Dateiname, hier php.m, müssen übereinstimmen. Den Programmcode können Sie direkt in das Editorfenster kopieren und speichern.

Zur Berechnung der Wurzeln beliebiger Polynome mit Octave muss man diese nur in die Standardform

$$a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \ldots + a_1 \cdot x + a_0 = 0 \quad (5)$$

bringen, die Koeffizienten als Vektor, hier v, übergeben und die Funktion roots liefert die Lösungen des Polynoms als Vektor r. Zu Einzelheiten der Programmierung sei auf das Handbuch verwiesen.

Bei pH-Wert Berechnungen kann nur eine der n Lösungen die gesuchte sein. Bei einem richtig formulierten Polynom ist dies die einzige reelle Lösung > 0 . Die for Schleife sucht diese heraus, berechnet den pH und return übergibt das Ergebnis als Rückgabewert der Funktion.

Im Befehlsfenster kann man nun die Funktion mit konkreten Werten für die Konzentrationen und den pK_a aufrufen, zum Beispiel für einen 0,1 M Acetatpuffer:

```
>> php(.1, .1, 4.75)
   ans =  4.7502
```

und erhält den pH im Standardformat mit 5 gültigen Ziffern.

Schöner und sinnvoller wird die Ausgabe mit einer Formatierung:

```
>> printf("pH = %.2f\n",
   php(.1, .1, 4.75))
pH = 4.75
```

Die Begrenztheit der Henderson-Hasselbalch Gleichung ist in Abbildung 3 mit einigen selbsterklärenden Beispielen dargestellt.

```
# php.m
# Berechnung des pH für Pufferlösungen
# [H+]³+(c0(MA)+Ka)[H+]²-(c0(HA) Ka+KW) [H+]-Ka·Kw = 0

function ph = php(c0ha, c0ma, pka)
        kw = 10^-14;
        ka = 10^-pka;
        a = 1;
        b = c0ma + ka;
        c = -ka * c0ha - kw;
        d = -ka * kw;
        v = [a,b,c,d];
        r = roots(v);
        for i=1:3
             if r(i)>0
                  ph = -log10(r(i));
                  return
             endif
        endfor
endfunction
```

Abb. 2: php-Programmcode für Gleichung (3)

```
>> printf("pH = %.2f\n", php(1e-4, 1e-4, 4.75))
pH = 4.87
>> printf("pH = %.2f\n", php(1e-7, 1e-7, 4.75))
pH = 6.79
>> printf("pH = %.2f\n", php(1e-2, 1e-2, 2))
pH = 2.38
>> printf("pH = %.2f\n", php(.001, .001, 10))
pH = 9.93
```

Abb.3: Beispiele für die Begrenztheit der Henderson-Hasselbalch-Gleichung. Mit zunehmender Verdünnung weichen die pH Werte für äquimolare Säure- und Salzkonzentrationen vom pK_a ab, und dies ist um so eher, je mehr der pK_a von 7 abweicht.

Natürlich bleibt die Berechnung des pH auch für starke Säuren

```
>> printf("pH = %.2f\n",
   php(.1, .1, -7))
      pH = 1.00
```

oder für Säuren allein

```
>> printf("pH = %.2f\n",
   php(.1, 0, 4.75))
  pH = 2.88
```

richtig.

Anschaulicher ist es, diese Abhängigkeiten grafisch darzustellen. In Octave gibt es hierfür die Funktion plot. Für die Erstellung eines zweidimensionalen Diagramms übergibt man plot die x- und y-Werte als Vektoren.

Um den pH eines Puffers als Funktion der Konzentration darzustellen, kann man die Funktion php in einer Schleife berechnen, speichert die Konzentrationen und Ergebnisse in Vektoren und ruft plot wie in Abbildung 4 dargestellt auf.

t wird als Vektor mit den Elementen 0 0,1 0,2 ... 8 vorgegeben. Die Laufvariable k berechnet mit den – hier 81 – Elementen von t die Konzentrationen als Zehnerpotenzen und die zugehörigen pH-Werte. plot trägt den pH gegen den Logarithmus der Konzentration auf.

Zu den umfangreichen Formatierungsmöglichkeiten von plot sei auf das Handbuch verwiesen.

Das Programm muss als Datei charlot.m zusammen mit php.m im Pfad von Octave liegen. Im Befehlsfenster kann man nach Aufruf von charlot den pK_s eingeben:

```
>> charlot
 pka: 4.75
```

Es öffnet sich ein Fenster[1] mit der Darstellung der Konzentrationsabhängigkeit des pH der Pufferlösung, hier wieder der Acetatpuffer. Wie in Abbildung 5 gezeigt, bleibt der pH bis knapp 10^{-4} mol/L Pufferkonzentration stabil und geht mit zunehmender Verdünnung richtig gegen 7.

Die grafische Ausgabe lässt sich über ihr Fenstermenü *File* in verschiedenen Formaten speichern.

Mehrere Kurven lassen sich nacheinander in das gleiche Fenster zeichnen (Abbildung 6). Hierzu gibt man zuerst den Befehl hold on[2] ein:

```
>> hold on
>> charlot
pka: 4.75
>> charlot
pka: 2
>> hold off
```

```
# charlot.m
# Grafische Ausgabe der Konzentrationsabhängigkeit
# eines Puffer-pH

#Löschen alter Werte
clear all

# t = Start:Schrittweite:Endpunkt für den
# Konzentrationsvektor c = 10^-t
t=0:.1:8;

# Initialisierung der Länge von k
k=1;

# Eingabe des pKa:
pka = input("pka: ");

while (k <= length(t))
      c(k) = 1*10^-t(k);
      pH(k) = php(c(k), c(k), pka);
      k++;
endwhile

plot (log10(c), pH, "linewidth", 5);
      set(gca, "linewidth", 3, "fontsize", 12);
      title ("pH von Pufferlösungen vs. Konzentration");
      xlabel ("log(c)", "fontsize", 20);
      ylabel ("pH", "fontsize", 20);
```

Abb. 4: php-Programmcode zur Darstellung des pH eines Puffers als Funktion der Konzentration

Die Pufferlösung mit dem pH 2 muss mindesten eine Konzentration von 0,1 M aufweisen, um den pH zu halten. Entsprechend empfindlich würde sie auf gegebenenfalls unbeabsichtigte Verdünnung mit einer deutlichen pH-Verschiebung reagieren.

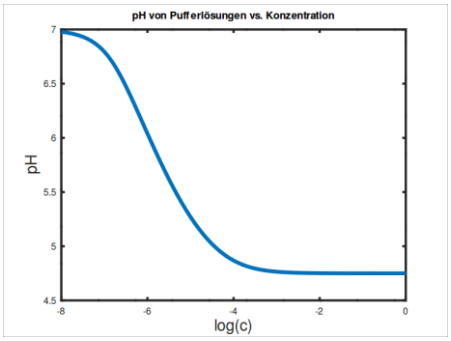

Abb. 5: Konzentrationsabhängigkeit des pH der Pufferlösung (Acetatpuffer)

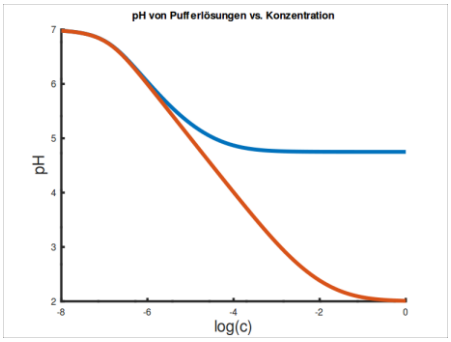

Abb. 6: Darstellung mehrerer Kurven

Anmerkung:

1) Das Fenster mit der Grafikausgabe gerät durch den Wechsel zum Editor in den Hintergrund.

2) Nach der Anwendung von hold gelingt das Speichern der Grafik nur mit dem Befehl print -dsvg grafik.svg .

Im alkalischen ergibt sich zum Beispiel für einen NH_4Cl-Puffer mit dem

$$pK_a = pK_W - pK_a = 9{,}25$$

für eine äquimolare NH_4Cl/NaOH Lösung die in Abbildung 7 dargestellte Abhängigkeit des Puffer-pH von der Pufferkonzentration.

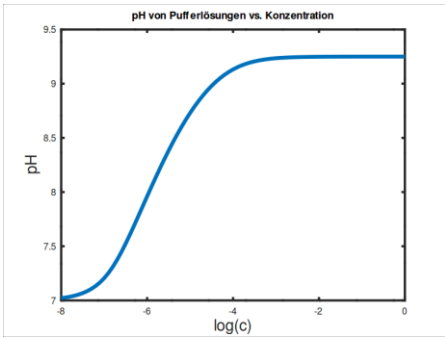

Abb. 7: Konzentrationsabhängigkeit des pH der Pufferlösung (NH_4Cl/NaOH)

```
function ph = phps(c0ha, c0ma, pka, chx, pkx)
       kw = 10^-14;
  ka = 10^-pka;
  kx = 10^-pkx;
  a = 1;
  b = ka + kx + c0ma;
  c = ka * (kx - c0ha) + kx * (c0ma - chx) - kw;
  d = -ka * kx * (c0ha + chx) + kw * (ka + kx);
  e = -ka * kx * kw;
  v = [a, b, c, d, e];
  r = roots(v);
  for i = 1:4;
    if r(i) > 0
      ph = -log10(r(i));
                    return
              endif
       endfor
endfunction
```

Abb. 8: php-Programmcode zur Berechnung der Gleichung (6)

Säurezusatz zu einer Pufferlösung

Herkömmlich rechnet man für den Zusatz einer Säure HX zu einem Puffer nach Gleichung (1a)

$$pH = pK_a + log\frac{c_0(MA) - c(HX)}{c_0(HA) + c(HX)} \quad (5)$$

die pH-Wert Änderung aus. Für Gleichung (5) gelten jedoch dieselben Beschränkungen wie für Gleichung (1).

Hier kommt zur Pufferlösung ein weiteres Säure-Base-Gleichgewicht hinzu. Damit wird die exakte Berechnung der Wasserstoff-Ionenkonzentration eine Gleichung vierten Grades in $[H^+]$ (Gleichung (6), unten stehend).

In Octave berechnet sich die Lösung dieser Gleichung mit der in Abbildung 8 aufgeführten Funktion.

Man kann `phps` mit numerischen Werten direkt im Befehlsfenster aufrufen, um den pH eines konkreten Puffer-Säure-Gemisches zu berechnen.

Interessanter ist die grafische Darstellung der Abhängigkeit des pH von der zugesetzten Säurekonzentration, wie in Abbildung 9 und 10 beschrieben.

Wie immer müssen sich `php_cx.m` und `phps.m` im Arbeitsverzeichnis von Octave befinden.

Dem Leser bleibt es überlassen, die Pufferkonzentration, auch unterschiedlich für c_0(HA) und c_0(MA), und den pK_a zu ändern und Säuren anderer Stärke zuzugeben. Die Abbildung 10 entspricht mit $pK_x = -7$ der Zugabe von Salzsäure.

Mit den gegebenen Skripten als Ausgangspunkt sollte es nicht schwer fallen, auch andere Abhängigkeiten grafisch darzustellen. Zum Beispiel, wie stark weicht für gegebene äquimolare Pufferkonzentrationen der pH eines Puffers von seinem pK_a ab?

Literatur

[1] Mortimer R. G.: Physical Chemistry, 3rd ed., Elsevier Academic Press, Burlington 2008

[2] Job G., Rüffler R.: Physikalische Chemie, Vieweg +Teubner Verlag | Springer Fachmedien, Wiesbaden 2011

[3] Hamann C. H., Hamnett A., Vielstich W.: Electrochemistry, 2nd ed., Wiley-VCh, Weinheim 2007

[4] Butler J. N.: Ionic Equilibrium, Addison-Wesley, Reading 1964

[5] Bliefert C.: pH-Wert-Berechnungen, VCh, Weinheim 1978

[6] de Levie R.: Chem. Educator 7 (2002) 132–135

[7] Bronstein I.N., Semendjajew K. A.: Taschenbuch der Mathematik, Teubner, Leipzig 1969

[8] Charlot, G.: Analytica Chimica Acta 1 (1947) 59–68

[9] Linge S., Langtangen H. P.: Programming for Computations – MATLAB/Octave, Springer Open, Heidelberg 2016

[10] GNU operating system

$$[H^+]^4 + (K_a + K_x + c_0(MA))\,[H^+]^3 + (K_a(K_x - c_0(HA)) + K_x(c_0(MA) - c(HX)) - K_W)\,[H^+]^2 - (K_a{\cdot}K_x(c_0(HA) + c(HX)) + K_W(K_a + K_x))\,[H^+] - K_a{\cdot}K_x{\cdot}K_W = 0 \quad (6)$$

```
# php_cx.m
# Grafische Ausgabe der Abhängigkeit
# eines Puffer-pH von zugesetzter Säure

#Löschen alter Werte
clear all

# t = Start:Schrittweite:Endpunkt für den
# Konzentrationsvektor c = 10^-t
t=0:.01:5;

# Initialisierung der Länge von k
k=1;

# Input
# pKa Puffer:
pka = input("pka: ");
# Eingabe der Pufferkonzentration:
c0 = input("c0: ");
# pKx Säurezusatz:
pkx = input("pkx: ");

# Verlaufsberechnung für einen äquimolaren Puffer
while (k <= length(t))
       c(k) = 1*10^-t(k);
       pH(k) = phps(c0, c0, pka, c(k), pkx);
       k++;
endwhile

# Ausgabe
plot (log10(c), pH, "linewidth", 4);
set(gca, "linewidth", 3, "fontsize", 12);
title  ("pH von Pufferlösungen vs. Konzentration");
xlabel ("log(c(HX))", "fontsize", 20);
ylabel ("pH", "fontsize", 20);
l = legend (["Pufferkonzentration " num2str(c0, "%.2f") " mol/L"],...
            "location","southwest");
set (l, "fontsize", 20);
legend boxoff;
a = annotation('textbox',[.2 .5 .3 .3],'String',["pKx = " num2str(pkx,...
               "%.2f")], 'FitBoxToText','on','linestyle','none');
set (a, "fontsize", 20);
grid on;
```

Abb. 9: php-Programmcode zur grafischen Darstellung der Abhängigkeit des pH von der zugesetzten Säurekonzentration

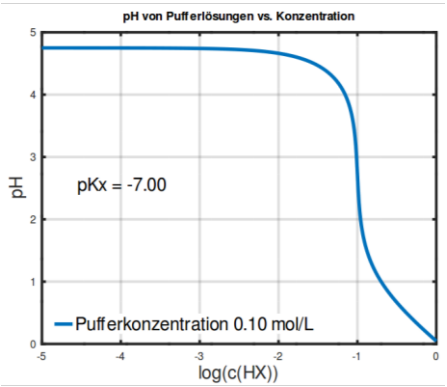

Abb. 10: grafische Darstellung der Abhängigkeit des pH von der zugesetzten Säurekonzentration (Salzsäure)